

0. 컴파일 및 실행 방법

1. ls 명령어를 쳤을 때 udp package를 확인할 수 있게 압축을 해제한다.

- javac -d . udp/*.java

A terminal window showing the directory structure and compilation command. The terminal shows:

```
ohsuyeong@ohsuyeong-MacBookAir:~/IntelliJ/computerNetwork/src
ls
udp
javac -d . udp/*.java
```

2. udp는 패키지 명이며 UDPp2pChatRoom이 메인 클래스이름이고 뒤에 포트번호를 입력하여 실행한다.

- java udp.UDPp2pChatRoom 1234(port number 지정)

A terminal window showing the execution command. The terminal shows:

```
java udp.UDPp2pChatRoom 1234
```

• 이 화면이 나오면 준비가 완료된 상태이다

1. main class : UDPp2pChatRoom

send 쓰레드와 receive쓰레드로 나누어 실행시켰고, receive쓰레드에서는 send쓰레드에서 받은 패킷을 출력해줘서 채팅방에 들어온 모든 peer가 그 내용을 알 수 있다.

```
package udp;

import java.net.*;
import java.io.*;

import static udp.HashTest.getSHA256;

public class UDPp2pChatRoom {

    public static void main(String[] args) throws IOException {
        // portNo 받아서 multiset 만들기
        int portNo = Integer.parseInt(args[0]);
        MulticastSocket multicastSocket = new MulticastSocket(portNo);

        Thread receiveThread = new Thread(new ReceiveThread(multicastSocket));
        receiveThread.start();

        Thread sendThread = new Thread(new SendThread(multicastSocket, portNo));
        sendThread.start();
    }
}
```

2. send thread : SendThread

1. 맨 처음 실행 시 #JOIN arg1 arg2의 형태가 아니면 다시 입력하도록 했다.

2. EXIT 후에도 다른 채팅방에 들어갈 수 있도록 하기 위해서 isJoin flag를 두었고, run()은 가장 크게 무한 while 문으로 돌려서 #QUIT을 입력했을 때만 실행을 종료할 수 있다.
- 여러 채팅방을 생성한 후 왔다갔다 할 수 있다.

```
// join 상태 아닐 때만
if (!isJoin) {
    BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));

    String sentence = null;
    try {
        sentence = inFromUser.readLine();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }

    String[] cmd = sentence.split(" ");

    if(sentence.equals("#QUIT")) {
        exit(0);
    }
    // #JOIN으로 시작하도록

    try {
        while (!cmd[0].equals("#JOIN") || cmd[2]==null) {
            System.out.println("[ERROR] Wrong command : Cannot start chatting room");
            sentence = inFromUser.readLine();
            cmd = sentence.split(" ");
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("[ERROR] Too few arguments : Cannot start chatting room");
        try {
            sentence = inFromUser.readLine();
        } catch (IOException ex) {
            throw new RuntimeException(ex);
        }
        cmd = sentence.split(" ");
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
}
```

3. JOIN 명령어와 인자들이 잘 들어왔다면 공백으로 split 하여 room과 peer의 이름을 받고, SHA-256 알고리즘을 이용해 IPAddress를 구해서 joinGroup해준다.

1. md = MessageDigest.getInstance("SHA-256"); 을 이용했고, 음수가 나오는 것은 & 0xff를 통해 양수로 바꿔준다.
2. 마지막 세 바이트를 각각 x, y, z로 string을 만들어주고, 이로 부터 InetAddress를 만들어 joinGroup의 인자로 넣어준다.

```
roomName = cmd[1];
peerName = cmd[2];

byte[] digest = getSHA256(roomName);
String IP = getIPAddress(digest);

InetAddress = InetAddress.getByName(IP);
multicastSocket.joinGroup(IPAddress);
```

4. 성공적으로 들어왔다면 해당 채팅방에 있던 다른 peer들도 누가 들어왔는지 알 수 있게 패킷을 전송한다.

5. 한 번 while을 돋 후 대화 메세지를 보내기 위해 join 부분을 건너뛰어야 하므로 isJoin을 true로 해준다.

```
sentence = "\"" + peerName + "\"" + " enters the [ " + roomName + " ] chatting room";
sendData = sentence.getBytes();

packet = new DatagramPacket(sendData, sendData.length, IPAddress, portNo);
multicastSocket.send(packet);

isJoin = true;
```

6. CASE 1) EXIT 입력

1. 누가 나갔는지 알려주기 위해 패킷을 전송해주고, leaveGroup으로 나온 후
2. IsJoin 을 false로 바꿔 다른 채팅방에 또 들어갈 수 있다.

```
// 사용자 입력값 받기 (#EXIT or #QUIT or sendData)
String sentence = null;
try {
    sentence = inFromUser.readLine();
} catch (IOException e) {
    throw new RuntimeException(e);
}

// #으로 시작할 때 - #EXIT or #QUITif (sentence.charAt(0) == '#') {
if (sentence.equals("#EXIT")) {
    sentence = "\"" + peerName + "\"" + " leaves the [ " + roomName + " ] chatting room";
    sendData = sentence.getBytes();

    packet = new DatagramPacket(sendData, sendData.length, IPAddress, portNo);

    try {
        multicastSocket.send(packet);
    }
```

```

        multicastSocket.leaveGroup(IPAddress);

        isJoin = false;
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
else {
    System.out.println("[ERROR] only #EXIT possible");
}
}

```

7. CASE 2) 대화 메세지 입력 (영어로 입력한다)

1. data의 크기가 512보다 작거나 같을 때 - 패킷을 바로 전송한다
2. data의 크기가 512보다 클 때 - offset과 여태까지 보낸 바이트와 보내야하는 바이트를 계산하여 쪼개서 보낸다(while로 처리)

```

// 채팅 보낼 때 - sendData else {
sentence = "Peer " + peerName + " : " + sentence;
sendData = sentence.getBytes();
// data 크기 512보다 작거나 같을 때
if (sendData.length <= 512) {
    packet = new DatagramPacket(sendData, sendData.length, IPAddress, portNo);

    try {
        multicastSocket.send(packet);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
// data 크기 512보다 클 때
else {
    int offset = 0;
    int bytesSent = 0;
    packet = new DatagramPacket(sendData, offset, 512, IPAddress, portNo);

    while (bytesSent < sendData.length) {
        try {
            multicastSocket.send(packet);

            bytesSent += packet.getLength();
            int bytesToSend = sendData.length - bytesSent;
            int size = (bytesToSend > 512) ? 512 : bytesToSend;

            packet.setData(sendData, bytesSent, size);

        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
}

```

3. receive thread : ReceiveThread

1. 데이터를 받을 공간을 만들어준다
2. receive를 통해 받아오고, 그 데이터의 길이 만큼 잘라준 후 출력해준다.

```

package udp;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.MulticastSocket;

public class ReceiveThread implements Runnable{
    private MulticastSocket multicastSocket;
    private byte[] receiveData = new byte[512];

    public ReceiveThread(MulticastSocket multicastSocket) {
        this.multicastSocket = multicastSocket;
    }

    @Override
    public void run() {
        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

        while (true) {
            try {
                multicastSocket.receive(receivePacket);
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            String sentence = new String(receivePacket.getData()).substring(0, receivePacket.getLength());
            System.out.println(sentence);
        }
    }
}

```

4. HashTest

입력으로부터 받은 roomName을 byte로 바꾸어서 반환한다.

```
package udp;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class HashTest {

    public static byte[] getSHA256(String str) {
        MessageDigest md = null;

        try {
            md = MessageDigest.getInstance("SHA-256");
            md.update(str.getBytes());
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        return md.digest();
    }
}
```

5. 실행 예시

0. 프로그램 실행 및 채팅방 시작

1. ls 를 쳤을 때 udp 패키지가 나오게 압축을 해제한다



2. javac -d . udp/*.java 를 친 후 다섯 개의 창 모두 java udp.UDPp2pChatRoom 9876를 쳐서 채팅방을 만든다

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
```

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
```

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
```

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
```

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
```

위의 화면이 나오면 채팅방이 모두 준비되었다.

1. 512바이트보다 작은 값이 잘 전달되는지 + 다른 peer에게도 메세지가 잘 보이는지

1-1) 1, 2, 3번 창에 순서대로 #JOIN cnet A , #JOIN cnet B, #JOIN cnet C를 입력한다

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
#JOIN cnet A
"A" enters the [ cnet ] chatting room
"B" enters the [ cnet ] chatting room
"C" enters the [ cnet ] chatting room
```

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
#JOIN cnet B
"B" enters the [ cnet ] chatting room
"C" enters the [ cnet ] chatting room
```

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
#JOIN cnet C
"C" enters the [ cnet ] chatting room
```

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
```

```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
java udp.UDPp2pChatRoom 9876
ls
udp
javac -d . udp/*.java
java udp.UDPp2pChatRoom 9876
```

1-2) A, B, C가 각각 할 말을 하면 각 채팅방에서 모두 메세지가 잘 출력되는 것을 볼 수 있다.

```

iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help

```

java udp.UDPp2pChatRoom 9876

java udp.UDPp2pChatRoom 9876

java udp.UDPp2pChatRoom 9876

java udp.UDPp2pChatRoom 9876

```

ls
udp
> javac -d . udp/*.java
> java udp.UDPp2pChatRoom 9876
#JOIN cnet A
"A" enters the [ cnet ] chatting room
"B" enters the [ cnet ] chatting room
"C" enters the [ cnet ] chatting room
hello my name is A
Peer A : hello my name is A
Peer B : nice to meet you A, I'm B
Peer C : hi A and B, I'm C

```

```

ls
udp
> javac -d . udp/*.java
> java udp.UDPp2pChatRoom 9876
#JOIN cnet B
"B" enters the [ cnet ] chatting room
"C" enters the [ cnet ] chatting room
Peer A : hello my name is A
nice to meet you A, I'm B
Peer B : nice to meet you A, I'm B
Peer C : hi A and B, I'm C

```

```

ls
udp
> javac -d . udp/*.java
> java udp.UDPp2pChatRoom 9876
#JOIN cnet C
"C" enters the [ cnet ] chatting room
Peer A : hello my name is A
Peer B : nice to meet you A, I'm B
hi A and B, I'm C
Peer C : hi A and B, I'm C

```

```

ls
udp
> javac -d . udp/*.java
> java udp.UDPp2pChatRoom 9876
#JOIN cnet D
"D" enters the [ cnet ] chatting room
"Peer A" : hello my name is A
"Peer B" : nice to meet you A, I'm B
"Peer C" : hi A and B, I'm C

```

1-3) A가 사용중인 1번 채팅방에 #EXIT 를 입력하면 A가 나갔음을 모든 채팅방에서 확인할 수 있다.

```

iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help

```

java udp.UDPp2pChatRoom 9876

java udp.UDPp2pChatRoom 9876

java udp.UDPp2pChatRoom 9876

java udp.UDPp2pChatRoom 9876

```

ls
udp
> javac -d . udp/*.java
> java udp.UDPp2pChatRoom 9876
#JOIN cnet A
"A" enters the [ cnet ] chatting room
"B" enters the [ cnet ] chatting room
"C" enters the [ cnet ] chatting room
hello my name is A
Peer A : hello my name is A
Peer B : nice to meet you A, I'm B
Peer C : hi A and B, I'm C
bye guys, I'm leaving
Peer A : bye guys, I'm leaving
#EXIT
"A" leaves the [ cnet ] chatting room

```

```

ls
udp
> javac -d . udp/*.java
> java udp.UDPp2pChatRoom 9876
#JOIN cnet B
"B" enters the [ cnet ] chatting room
"C" enters the [ cnet ] chatting room
Peer A : hello my name is A
nice to meet you A, I'm B
Peer B : nice to meet you A, I'm B
Peer C : hi A and B, I'm C
Peer A : bye guys, I'm leaving
"A" leaves the [ cnet ] chatting room

```

```

ls
udp
> javac -d . udp/*.java
> java udp.UDPp2pChatRoom 9876
#JOIN cnet C
"C" enters the [ cnet ] chatting room
Peer A : hello my name is A
Peer B : nice to meet you A, I'm B
hi A and B, I'm C
Peer C : hi A and B, I'm C
Peer A : bye guys, I'm leaving
"A" leaves the [ cnet ] chatting room

```

```

ls
udp
> javac -d . udp/*.java
> java udp.UDPp2pChatRoom 9876
#JOIN cnet D
"D" enters the [ cnet ] chatting room
"Peer A" : hello my name is A
"Peer B" : nice to meet you A, I'm B
"Peer C" : hi A and B, I'm C

```

2. 512바이트보다 큰 값이 잘 전달되는지 + 다른 peer에게도 메세지가 잘 보이는지

2-1) B가 있는 2번 채팅방에 2000바이트짜리 문자를 보냈을 때 현재 연결되어있는 C의 채팅방에도 이 메세지가 잘 전달됨을 볼 수 있다.

The screenshot shows four terminal windows (iTerm2) running on a Mac OS X desktop. Each window has a title bar indicating it's a Java UDP-based chat room (9876). The windows are arranged in a 2x2 grid.

- Window 1 (Top Left):** Shows Peer A's session. It receives messages from Peer B and Peer C, and sends its own messages.
- Window 2 (Top Right):** Shows Peer B's session. It receives messages from Peer A and Peer C, and sends its own messages.
- Window 3 (Bottom Left):** Shows Peer C's session. It receives messages from Peer A and Peer B, and sends its own messages.
- Window 4 (Bottom Right):** Shows Peer D's session. It receives messages from Peer A, Peer B, and Peer C, and sends its own messages.

In all windows, the messages are clearly visible and correctly received by the intended peers, demonstrating the successful transmission of 2000-byte messages between connected clients.

3. 채팅방을 여러 개 만들고 그 안에서 각각 송수신이 잘 일어나는지

3-1) 4, 5번 창은 각각 #JOIN hyu D와 #JOIN hyu E를 쳐서 hyu 채팅방으로 들어온다.

The screenshot shows six terminal windows (iTerm2) running on a Mac OS X desktop, illustrating the creation of multiple chat rooms (hyu) and their separate communication.

- Windows 1-4 (Left Column):** These windows represent hyu D. They show Peer A joining the room, sending and receiving messages from Peers B and C.
- Windows 5-6 (Right Column):** These windows represent hyu E. They show Peer A joining the room, sending and receiving messages from Peers B and C.

The windows clearly demonstrate that multiple rooms can coexist and operate independently, with each room having its own distinct set of participants and message flow.

3-2) 2, 3번의 B와 C 그룹과 4, 5번의 D, E그룹이 따로 채팅을 진행할 수 있다. 다음은 B와 C, D와 E가 각각의 채팅방에서 이야기를 한 후 모습이다.

The image shows four terminal windows (labeled 2, 3, 4, 5) running the Java UDP-based chat application. Terminal 2 and 4 show the initial setup and communication between Peers A, B, and C. Terminal 3 and 5 show the initial setup and communication between Peers D and E. All peers are using port 9876.

```
Terminal 2 (Peers A, B, C):  
ls  
java udp.UDPP2pChatRoom 9876  
> java -d . udp/*.java  
> java udp.UDPP2pChatRoom 9876  
#JOIN cnet A  
"A" enters the [ cnet ] chatting room  
"B" enters the [ cnet ] chatting room  
"C" enters the [ cnet ] chatting room  
hello my name is A  
Peer A : hello my name is A  
Peer B : nice to meet you A, I'm B  
Peer C : hi A and B, I'm C  
bye guys, I'm leaving  
Peer A : bye guys, I'm leaving  
#EXIT  
"A" leaves the [ cnet ] chatting room  
[]  
  
Terminal 3 (Peers D, E):  
ls  
java udp.UDPP2pChatRoom 9876  
> java -d . udp/*.java  
> java udp.UDPP2pChatRoom 9876  
#JOIN cnet A  
"D" enters the [ hyu ] chatting room  
"E" enters the [ hyu ] chatting room  
hi E, this is hyu chatting room  
Peer D : hi E, this is hyu chatting room  
Peer E : oh, how are you D, I love hanyang university  
[]  
  
Terminal 4 (Peers A, B, C):  
ls  
java udp.UDPP2pChatRoom 9876  
> java -d . udp/*.java  
> java udp.UDPP2pChatRoom 9876  
#JOIN cnet A  
"A" enters the [ cnet ] chatting room  
"B" enters the [ cnet ] chatting room  
"C" enters the [ cnet ] chatting room  
hello my name is A  
Peer A : hello my name is A  
Peer B : nice to meet you A, I'm B  
Peer C : hi A and B, I'm C  
bye guys, I'm leaving  
Peer A : bye guys, I'm leaving  
#EXIT  
"A" leaves the [ cnet ] chatting room  
# JOIN cnet A  
[ERROR] Wrong command : Cannot start chatting room  
#JOIN cnet A  
"A" enters the [ cnet ] chatting room  
[]  
  
Terminal 5 (Peers D, E):  
ls  
java udp.UDPP2pChatRoom 9876  
> java -d . udp/*.java  
> java udp.UDPP2pChatRoom 9876  
#JOIN cnet A  
"D" enters the [ hyu ] chatting room  
"E" enters the [ hyu ] chatting room  
hi E, this is hyu chatting room  
Peer D : hi E, this is hyu chatting room  
Peer E : oh, how are you D, I love hanyang university  
[]
```

4. 하나의 채팅방을 나오고 원래 있던 채팅방에 들어가서 잘 통신이 되는지

4-1) A를 다시 cnet에 들어보낸다 #JOIN cnet A 채팅을 보내본다. (이상한 명령어가 들어갔을 때 오류가 남을 알 수 있다. A가 다시 잘 들어왔다.)

The image shows four terminal windows (labeled 2, 3, 4, 5) running the Java UDP-based chat application. Terminal 2 and 4 show the initial setup and communication between Peers A, B, and C. Terminal 3 and 5 show the initial setup and communication between Peers D and E. All peers are using port 9876.

```
Terminal 2 (Peers A, B, C):  
ls  
java udp.UDPP2pChatRoom 9876  
> java -d . udp/*.java  
> java udp.UDPP2pChatRoom 9876  
#JOIN cnet A  
"A" enters the [ cnet ] chatting room  
"B" enters the [ cnet ] chatting room  
"C" enters the [ cnet ] chatting room  
hello my name is A  
Peer A : hello my name is A  
Peer B : nice to meet you A, I'm B  
Peer C : hi A and B, I'm C  
bye guys, I'm leaving  
Peer A : bye guys, I'm leaving  
#EXIT  
"A" leaves the [ cnet ] chatting room  
# JOIN cnet A  
[ERROR] Wrong command : Cannot start chatting room  
#JOIN cnet A  
"A" enters the [ cnet ] chatting room  
[]  
  
Terminal 3 (Peers D, E):  
ls  
java udp.UDPP2pChatRoom 9876  
> java -d . udp/*.java  
> java udp.UDPP2pChatRoom 9876  
#JOIN cnet A  
"D" enters the [ hyu ] chatting room  
"E" enters the [ hyu ] chatting room  
hi E, this is hyu chatting room  
Peer D : hi E, this is hyu chatting room  
Peer E : oh, how are you D, I love hanyang university  
[]  
  
Terminal 4 (Peers A, B, C):  
ls  
java udp.UDPP2pChatRoom 9876  
> java -d . udp/*.java  
> java udp.UDPP2pChatRoom 9876  
#JOIN cnet A  
"A" enters the [ cnet ] chatting room  
"B" enters the [ cnet ] chatting room  
"C" enters the [ cnet ] chatting room  
hello my name is A  
Peer A : hello my name is A  
Peer B : nice to meet you A, I'm B  
Peer C : hi A and B, I'm C  
bye guys, I'm leaving  
Peer A : bye guys, I'm leaving  
#EXIT  
"A" leaves the [ cnet ] chatting room  
# JOIN cnet A  
[ERROR] Wrong command : Cannot start chatting room  
#JOIN cnet A  
"A" enters the [ cnet ] chatting room  
[]  
  
Terminal 5 (Peers D, E):  
ls  
java udp.UDPP2pChatRoom 9876  
> java -d . udp/*.java  
> java udp.UDPP2pChatRoom 9876  
#JOIN cnet A  
"D" enters the [ hyu ] chatting room  
"E" enters the [ hyu ] chatting room  
hi E, this is hyu chatting room  
Peer D : hi E, this is hyu chatting room  
Peer E : oh, how are you D, I love hanyang university  
[]
```

4-2) A가 말을 하면 B, C에게도 잘 전달된다.

```

Terminal 1 (Peer A):
ls
java udp.UDPP2pChatRoom 9876
> java -d . udp/*.java
> java udp.UDPP2pChatRoom 9876
#JOIN cnet A
"A" enters the [ cnet ] chatting room
"B" enters the [ cnet ] chatting room
"C" enters the [ cnet ] chatting room
hello my name is A
Peer A : hello my name is A
Peer B : nice to meet you A, I'm B
Peer C : hi A and B, I'm C
bye guys, I'm leaving
Peer A : bye guys, I'm leaving
#EXIT
"A" leaves the [ cnet ] chatting room
#1 JOIN cnet A
[ERROR] Wrong command : Cannot start chatting room
#JOIN cnet A
"A" enters the [ cnet ] chatting room
I'm back again
Peer A : I'm back again
I missed you guys
Peer A : I missed you guys

Terminal 2 (Peer B):
ls
java udp.UDPP2pChatRoom 9876
> java udp.UDPP2pChatRoom 9876
#JOIN hyu B
"D" enters the [ hyu ] chatting room
"E" enters the [ hyu ] chatting room
hi E, this is hyu chatting room
Peer D : hi E, this is hyu chatting room
Peer E : oh, how are you D, I love hanyang university
Peer E : oh, how are you D, I love hanyang university

Terminal 3 (Peer C):
ls
java udp.UDPP2pChatRoom 9876
> java udp.UDPP2pChatRoom 9876
#JOIN hyu C
"D" enters the [ hyu ] chatting room
"E" enters the [ hyu ] chatting room
hi E, this is hyu chatting room
Peer D : hi E, this is hyu chatting room
Peer E : oh, how are you D, I love hanyang university
Peer E : oh, how are you D, I love hanyang university

Terminal 4 (Peer D):
ls
java udp.UDPP2pChatRoom 9876
> java udp.UDPP2pChatRoom 9876
#JOIN hyu D
"E" enters the [ hyu ] chatting room
Peer E : oh, how are you D, I love hanyang university
Peer E : oh, how are you D, I love hanyang university

```

4-3) A가 cnet에서 나온 후 hyu에 들어가보자. A가 들어갔음을 D, E에게도 전달한다. 이제 A, D, E가 같은 채팅방에 있다.

```

Terminal 1 (Peer A):
ls
java udp.UDPP2pChatRoom 9876
> java -d . udp/*.java
> java udp.UDPP2pChatRoom 9876
#JOIN cnet A
"A" enters the [ cnet ] chatting room
"B" enters the [ cnet ] chatting room
"C" enters the [ cnet ] chatting room
hello my name is A
Peer A : hello my name is A
Peer B : nice to meet you A, I'm B
Peer C : hi A and B, I'm C
bye guys, I'm leaving
Peer A : bye guys, I'm leaving
#EXIT
"A" leaves the [ cnet ] chatting room
#1 JOIN cnet A
[ERROR] Wrong command : Cannot start chatting room
#JOIN cnet A
"A" enters the [ cnet ] chatting room
I'm back again
Peer A : I'm back again
I missed you guys
Peer A : I missed you guys
#EXIT
Peer A : EXIT
#EXIT
#JOIN hyu A
"A" enters the [ hyu ] chatting room

Terminal 2 (Peer B):
ls
java udp.UDPP2pChatRoom 9876
> java udp.UDPP2pChatRoom 9876
#JOIN hyu B
"D" enters the [ hyu ] chatting room
"E" enters the [ hyu ] chatting room
hi E, this is hyu chatting room
Peer D : hi E, this is hyu chatting room
Peer E : oh, how are you D, I love hanyang university
Peer E : oh, how are you D, I love hanyang university

Terminal 3 (Peer C):
ls
java udp.UDPP2pChatRoom 9876
> java udp.UDPP2pChatRoom 9876
#JOIN hyu C
"D" enters the [ hyu ] chatting room
"E" enters the [ hyu ] chatting room
hi E, this is hyu chatting room
Peer D : hi E, this is hyu chatting room
Peer E : oh, how are you D, I love hanyang university
Peer E : oh, how are you D, I love hanyang university

Terminal 4 (Peer D):
ls
java udp.UDPP2pChatRoom 9876
> java udp.UDPP2pChatRoom 9876
#JOIN hyu D
"E" enters the [ hyu ] chatting room
Peer E : oh, how are you D, I love hanyang university
Peer E : oh, how are you D, I love hanyang university

```

4-4) A, D, E의 말이 서로에게 잘 전달되며 여러 채팅방을 돌아다니면서 채팅할 수 있음을 알 수 있다.

The image shows four iTerm windows running Java UDP-based peer-to-peer chat programs. The windows are labeled #2, #3, #4, and #6. Each window displays a sequence of messages from different peers (A, B, C, D, E) as they join and leave various chat rooms (cnet, hyu, cnet, hyu, etc.). The messages include greetings, room descriptions, and acknowledgments like 'I'm back again' or 'I missed you guys'. The communication is fluid, demonstrating that peers can move between rooms and still receive messages from others.

4-5) #EXIT을 한 후 #QUIT으로 프로그램을 종료할 수 있다.

This image shows two iTerm windows. The left window (#5) shows a peer named 'D' joining the 'hyu' room, sending messages like 'hi E, this is hyu chatting room'. The right window (#6) shows another peer named 'E' joining the same 'hyu' room. Both windows show a series of messages from peers A through E as they interact in the room. At the end of each session, the peer sends a '#QUIT' command to exit the room, which is then acknowledged by the server.

5. 예외 처리

맨 처음 실행한 후 #JOIN을 치지 않거나 인자가 부족한 경우 에러메세지를 출력하게 했고, 채팅방에 들어온 후에는 #EXIT만 가능하게 했다.

This image shows an iTerm window where a user has entered invalid commands. The user types '#JOIN hyu D' but forgets to include the room name 'hyu'. The program responds with an error message: '[ERROR] Wrong command : Cannot start chatting room'. Another attempt with '#JOIN a' also fails with '[ERROR] Too few arguments : Cannot start chatting room'. The user then tries to enter a room with '#JOIN cnet A', but the program rejects it with '[ERROR] Only #EXIT possible'. Finally, the user exits the program with '#QUIT'.